ANES. Arreglos Neuronales basados en la Evolución de Subpoblaciones

Lic. Leonardo Corbalán[1], Lic. Laura Lanzarini[2], Ing. Armando E. De Giusti[3]

Instituto de Investigación en Informática LIDI[4]
Facultad de Informática. Universidad Nacional de La Plata.

Resumen

Existe cierto tipo de problemas cuya solución requiere del aprendizaje de estrategias. Esto es lo que se conoce como *Comportamiento Complejo* y se aplica generalmente a la robótica y al desarrollo de sistemas de control de procesos, cuya resolución no es directa.

Los arreglos neuronales evolutivos, a diferencia de los métodos convencionales basados en una única red neuronal, están conformados por un conjunto de redes que se organizan en forma de arreglo. Cada una de estas redes representa una parte de la solución esperada. Si bien han demostrado ser capaces de brindar soluciones eficientes, requieren de la división explícita del problema original en subtareas.

El presente artículo describe un nuevo método, ANES, que permite evolucionar subpoblaciones de redes, facilitando de esta manera la obtención de componentes especializadas sin requerir ningún tipo de información del problema específico a resolver.

Las mediciones realizadas del método propuesto, aplicado a problemas de evasión de obstáculos y recolección de objetos, muestran la superioridad de ANES con respecto a los métodos tradicionales que manejan poblaciones de redes neuronales. En particular se ha utilizado SANE como referente comparativo debido a su alto rendimiento.

Finalmente se presentan las conclusiones y se plantean algunas líneas de trabajo futuras.

Palabras claves: Redes Neuronales Evolutivas, Arreglos Neuronales Evolutivos, Aprendizaje, Algoritmos Genéticos. Subpoblaciones

---

[1] *Becario UNLP.III-LIDI. Auxiliar Docente. Fac. de Informática. UNLP. corbalan@lidi.info.unlp.edu.ar*
[2] *Profesor Titular DE. III-LIDI. Fac. de Informática. Universidad Nacional de La Plata. laural@info.unlp.edu.ar*
[3] *Investigador Principal del CONICET. Profesor Titular Dedicación Exclusiva. III-LIDI. Fac. de Informática. Universidad Nacional de La Plata. degiusti@info.unlp.edu.ar*
[4] *III-LIDI miembro del Instituto de Investigación en Ciencia y Tecnología Informática (IICyTI) - Facultad de Informática. UNLP - Calle 50 y 115 1er Piso, (1900) La Plata, Argentina. TE/Fax +(54) (221) 422-7707. http://lidi.info.unlp.edu.ar*

1. Introducción

Las Redes Neuronales Evolutivas (RNE) son un caso particular de las redes neuronales artificiales (RNA) en las cuales la adaptación se realiza por entrenamiento y, principalmente, por evolución [12]. La evolución se ha utilizado en diversas formas: para conseguir los pesos de conexión, el diseño de la arquitectura, el valor de los parámetros iniciales, las reglas de aprendizaje, etc. [13].

Freeman y Skapura [6] argumentaron la necesidad de aprender a combinar RNAs pequeñas, poniéndolas bajo el control de otras redes para resolver el problema del escalado. Xin Yao y Yong Liu [11] estudiaron los beneficios de utilizar la población completa de redes neuronales obtenidas en la última generación, producto de un proceso evolucionario, en lugar de únicamente la de mejor fitness. Más recientemente, Bruce y Miikkulainen [1], trabajando sobre el problema de reconocimiento de caracteres escritos a mano alzada, demostraron que todas las redes neuronales de una población, combinado con una técnica efectiva de especialización, pueden responder mejor de manera colectiva que cualquiera de ellas individualmente.

Bajo el lineamiento general de explorar soluciones basadas en múltiples redes neuronales, se presentaron en [4] los arreglos neuronales evolutivos (ANE). Este método demostró ser más eficiente que otras estrategias neuroevolutivas al combinar los beneficios de la evolución incremental con la potencia de varias redes neuronales integradas en un único controlador. Sin embargo, la necesidad de definir explícitamente los subobjetivos en cada subetapa, herencia obligada de la evolución incremental, constituye el punto más débil del método dificultando su generalización.

Una estrategia alternativa para evolucionar arreglos neuronales fue presentada en [5] a fin de evitar la definición explícita de subobjetivos. De esta manera es posible construir algoritmos evolutivos fácilmente adaptables a otros tipos de problemas. No obstante, las soluciones conseguidas suelen ser poco tolerantes a disturbios que alteren los tiempos normales del proceso.

En este artículo se propone un nuevo método de evolución de arreglos neuronales, ANES, que subsana en gran medida las dificultades mencionadas. ANES no trabaja en base a la definición explícita de subobjetivos y la estrategia de activación de las redes dentro del arreglo, no depende exclusivamente del paso del tiempo, sino que esencialmente es una función de la entrada de datos del controlador.

2. ANES

ANES permite obtener controladores formados por arreglos neuronales que resuelven problemas de control de procesos en forma más eficiente que las soluciones tradicionales. Las redes integrantes provenientes de la evolución de distintas subpoblaciones, aprenden a especializarse en distintas subtareas del proceso total a controlar. Así, del accionar coordinado de estas redes surge la resolución de un problema complejo en forma más eficiente.

2.1. Organización interna del controlador

El controlador está formado por un arreglo neuronal, es decir una n-upla de redes neuronales de la forma $C=(rn_1, rn_2, .., rn_n)$. Al igual que una red, un arreglo neuronal acepta una entrada de datos, se evalúa y produce la salida correspondiente. En cada instante t, la salida será provista por alguna de las redes integrantes. Sólo una de ellas permanece activa a la vez, por lo tanto el tiempo

de procesamiento del controlador será igual al de la red neuronal que se evalúa en dicho instante.

2.2. Funcionamiento del controlador

Al comienzo del proceso, la única red activa del controlador es rn_1 que resuelve todas las entradas hasta su auto-desactivación. Una vez desactivada, el control pasa a rn_2 que continúa evaluándose hasta que "decida" desactivarse, pasando el control a rn_3. Esta delegación de control prosigue hasta que eventualmente se activa rn_n quien permanece en dicho estado hasta la finalización del proceso. En todos los casos, la desactivación de las redes se efectiviza después de producida la salida del controlador, dejando activa la próxima red neuronal para el siguiente instante de evaluación.

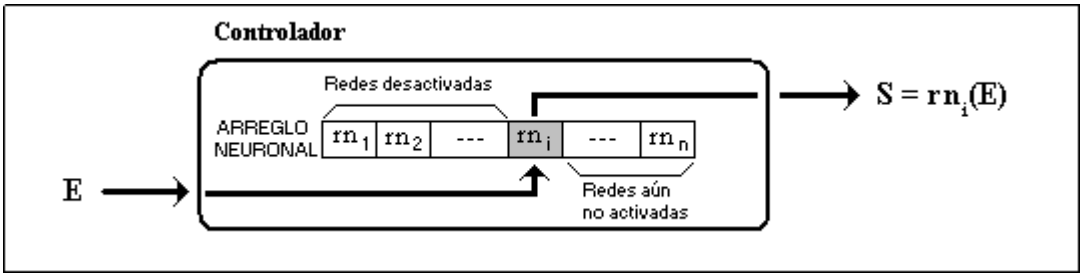


Figura 1. Estructura del controlador

Las redes del arreglo cuentan con una neurona de salida extra a la cantidad definida por el problema a resolver. El valor de dicha neurona no forma parte de la salida del controlador y se utiliza para determinar si la red evaluada debe desactivarse.

Sea $d_i(t)$ el valor de la neurona extra de salida de la red activa rn_i en el instante de evaluación t. La red neuronal rn_i seguirá activa para el instante $t+1$ si se cumple:

$$Abs(d_i(t)-d_i(t-1)) < c$$, donde c es una constante definida a priori.

Por el contrario, si no se cumple la condición anterior, rn_{i+1} se activará en el instante $t+1$.

De esta manera, el mecanismo de activación de las redes integrantes depende de la entrada de datos con que se estimule al controlador. No obstante, para restringir la posibilidad de que una red domine sobre las otras a lo largo del tiempo del proceso, se ha impuesto una restricción temporal disminuyendo el umbral conforme la red permanece activa. Sean T el tiempo total del proceso, n la dimensión del arreglo que conforma el controlador $(T >> n)$ y e_i la cantidad de evaluaciones de la red activa rn_i en el instante t, rn_i seguirá activa en el instante $t+1$ si se cumple:

$$Abs(d_i(t)-d_i(t-1)) < c-c.e_i.n.(1/T)$$

La expresión anterior asegura que cada red se evalúa al menos 1 vez y a lo sumo T/n veces. Dentro de este intervalo, el instante en que la red se desactiva depende exclusivamente de la entrada de datos.

3. Algoritmo evolutivo

Para obtener controladores de la forma $C=(rn_1, rn_2, .., rn_n)$ se evolucionan concurrentemente n poblaciones de redes, una para cada componente rn_i del controlador.

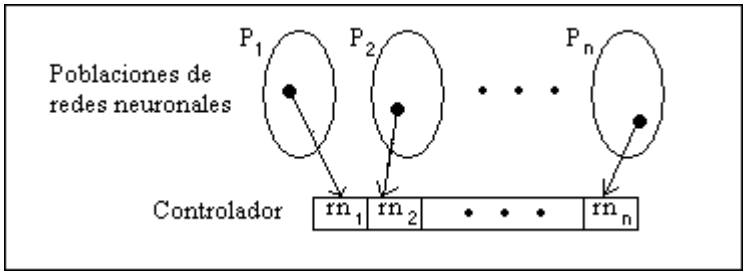


Figura 2. Subpoblaciones de redes neuronales

Las poblaciones *Pi* son evolucionadas de a una por vez, durante períodos de longitud variable medidos en cantidad de generaciones y determinados en función del fitness obtenido. Conforme avance el proceso, las evoluciones de las poblaciones se sucederán ordenadamente en una cola circular. Cada *Pi* eventualmente pasará por varios períodos de evolución.

El esquema general del algoritmo utilizado es el siguiente:

Generar aletoriamente n poblaciones de redes neuronales $P_1\ P_2 \ldots P_n$
i:=1
Mientras *no se alcance la solución al problema*
  Construir la población de controladores *neuronales* $P_C(i)$ **(*)**
  Evaluar cada controlador de $P_C(i)$ *asignando el fitness a los individuos de* P_i
  Si *(la curva del fitness es estacionaria)* **entonces**
    $i := i \bmod n + 1$
  Si no
    $P_i := próxima\_generación(P_i)$
  Fin si
Fin mientras

(*) Los controladores se construyen con cada una de las redes de la población sometida a evolución y la mejor red ranqueada según su fitness, en cada una de las restantes poblaciones. De esta forma, los controladores evaluados difieren entre sí, sólo en la i-ésima componente:

$$P_C(i)=\{(rn_1,\ldots,rn_i^1,\ldots rn_n),(rn_1,\ldots,rn_i^2,\ldots rn_n),\ldots(rn_1,\ldots,rn_i^k,\ldots rn_n)\}$$

Siendo rn_j la mejor red ranqueada en la población P_j, con $j<>i$.

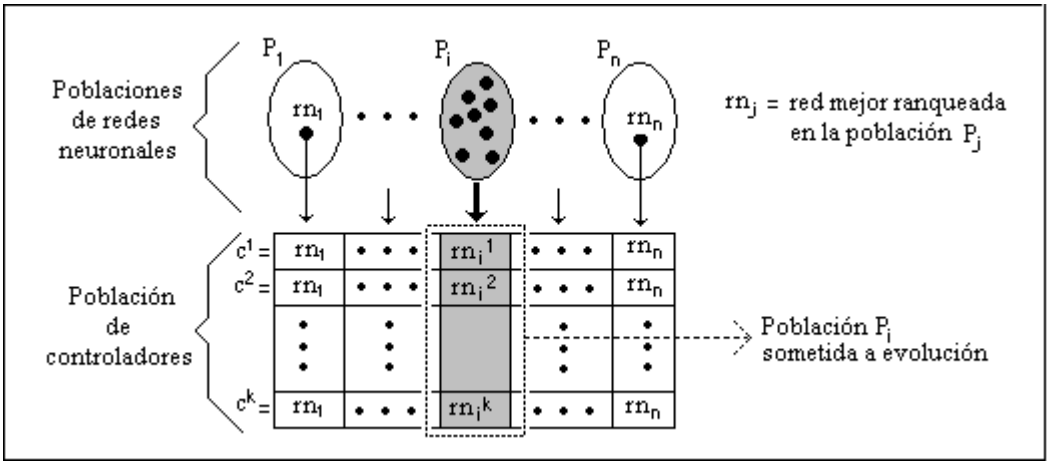


Figura 3. Evolución de las subpoblaciones

De esta manera, en cada período de evolución, existe una población P_i que intenta optimizar la integración de la componente rn_i del controlador con las restantes redes del mismo. Una vez que la curva del fitness obtenida durante la evaluación del controlador no mejora por algún número de generaciones previamente establecido, se pasa a evolucionar la próxima población iniciándose así un nuevo período evolutivo.

Durante la evaluación de los controladores, necesaria para el cálculo del fitness, no se activará ninguna red que no haya sido, o esté siendo, sometida a evolución. Así los controladores serán completamente funcionales luego del n-ésimo período de evolución. De no ser así, en el i-ésimo período evolutivo se estaría buscando la rn_i que mejor se integre con rn_{i+1} ,...,rn_n todas redes aleatorias aún no evolucionadas que conducirían a un pobre desempeño.

No se asume ninguna restricción sobre los parámetros de la red sometidos a evolución (pesos de conexión, arquitectura, función de transferencia, etc.). Puede hallarse un conjunto de variantes en [13] donde se citan varias investigaciones en neuroevolución incluyendo hibridaciones con algoritmos de aprendizaje tradicionales.

Para la obtención de la próxima generación en la población P_i de redes neuronales, no se asume ningún algoritmo evolutivo específico pudiendo incluso utilizarse un algoritmo genético simple como el presentado por Goldberg [7]. En particular, en el presente trabajo se ha utilizado SANE [8] [9] [10] que se describe brevemente pues también ha sido utilizado como referente comparativo en los testeos realizados.

Debido a que las subpoblaciones se evolucionan por turnos de a una a la vez, la carga computacional del algoritmo aquí presentado es similar a la de cualquier método convencional. Además, una buena implementación de la fase de evaluación de los controladores, permite mejorar los tiempos de procesamiento pudiendo, en algunos casos, utilizar una misma salida de una red neuronal para la población completa, ahorrando así tiempo de cómputo.

Una pequeña variante al algoritmo aquí presentado permite reducir los requerimientos de almacenamiento a niveles similares a los métodos convencionales. En lugar de mantener las subpoblaciones completas se conserva sólo un determinado porcentaje de las redes mejor ranqueadas en las mismas. Una vez obtenido nuevamente el turno para evolucionarse, la subpoblación es completada con cromosomas aleatorios que aportan diversidad genética

4. SANE

SANE ha demostrado las ventajas de la coevolución cooperativa en la búsqueda de soluciones a problemas de control, evidenciándose superior a las estrategias tradicionales.

En las soluciones convencionales que evolucionan redes neuronales, cada individuo de la población representa una red neuronal completa. Esto no ocurre en SANE donde los individuos de una población de neuronas son combinados para formar las redes que expresan la solución buscada. Esta combinación se somete a evolución en una población de “blueprints”.

Cada miembro de la población de neuronas codifica, con alfabeto binario, una secuencia de conexiones (*etiqueta:peso*) que definen completamente un nodo oculto de una red neuronal feedforward de una sola capa oculta. El campo *etiqueta* identifica la neurona de entrada o de salida con quien se establece la conexión. Cada miembro de la población de blueprints consiste en una

serie de punteros a la otra población que identifican las neuronas que construyen la red neuronal (Figura 3). La población de blueprints utiliza alfabeto real.

El algoritmo evolutivo trabaja construyendo las redes neuronales a partir de cada cromosoma blueprint. El fitness de cada red, obtenido por el desempeño de la misma en la resolución del problema planteado, es asignado al cromosoma blueprint. El fitness de los individuos de la población de neuronas se calcula como la suma de los cinco mejores fitness obtenidos en las redes en las que hayan participado. Se obtiene la próxima generación en la población de neuronas y luego la próxima generación en la población de blueprints.

SANE utiliza una estrategia de selección y reemplazo elitista en ambas poblaciones. La mitad mejor ranqueada de la población pasa a la próxima generación. El mejor cuarto es seleccionado para reproducirse completando con su descendencia la mitad restante (Figura 5).

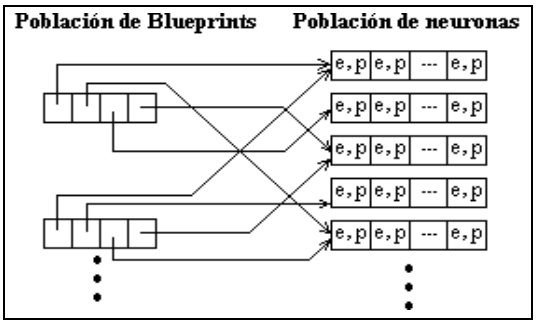


Figura 4. Poblaciones de SANE

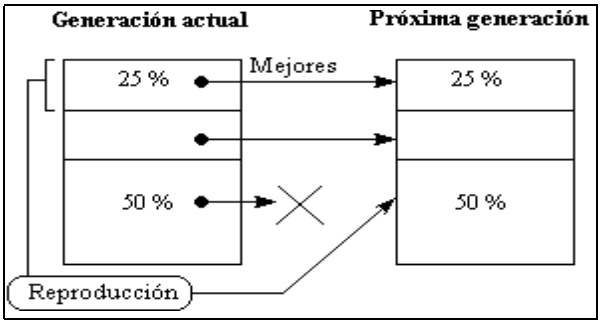


Figura 5. Selección en reemplazo en SANE

En cuanto a la reproducción, se utiliza crossover de un punto para conseguir el primer descendiente mientras que el segundo se obtiene por copia de uno de los parentales. En la población de neuronas se aplica mutación binaria con probabilidad 0.001 por bit. En la población de blueprints se aplican dos tipos de mutaciones: i) cambio de puntero a otra unidad de la población de neuronas elegida al azar con probabilidad 0.01 y ii) cambio de puntero a un descendiente de la unidad apuntada con probabilidad 0.5.

Puede consultarse [8][9][10] para una explicación más detallada de este método.

5. Evasión de obstáculos, búsqueda y recolección de objetos

5.1. Definición del problema

Se pretende conseguir comportamiento inteligente en un agente que se mueve libremente en dos dimensiones dentro de los límites de un entorno virtual, interactuando con obstáculos, que debe aprender a evitar, y objetos que debe encontrar y recoger.

Un controlador formado por un arreglo de redes neuronales dirige los movimientos del agente en un intervalo temporal simulado por la sucesión de n instantes discretos de tiempo (pasos de simulación). En cada instante el controlador es estimulado por un conjunto de señales de entrada. La salida está conformada por un par ordenado (x,y) que determina el ángulo de giro y desplazamiento que realiza el agente sobre la superficie. No obstante, los obstáculos presentes y los límites del entorno pueden impedir que el movimiento se lleve acabo.

El objetivo del problema es controlar al agente para que, partiendo desde un sitio determinado, encuentre y retire del escenario dos objetos, utilizando la menor cantidad de pasos de simulación posible. Al comenzar la simulación sólo se encuentra el objeto 1 en la posición indicada (figura 7). El objeto 2 se coloca en el escenario en el instante en que el objeto 1 desaparece al ser recogido por el agente.

5.2. Agente

Cada agente posee 13 sensores que trabajan de la siguiente forma (Figura 6):

- 5 sensores para detectar obstáculos, a corta distancia (2 veces su propio diámetro), distribuidos uniformemente hacia el frente del agente, para lograr un ángulo de 144º de visión.
- 4 sensores, uno para cada cuadrante, afectados a la detección del objeto 1.
- 4 sensores, uno para cada cuadrante, afectados a la detección del objeto 2.

Los sensores de obstáculos pueden considerarse como prolongaciones sensibles al tacto orientadas hacia delante en cinco direcciones, como se aprecia en la parte izquierda de la figura 6. Proporcionan un valor real del intervalo [0,1] directamente proporcional a la cercanía del obstáculo detectado. En cambio, los detectores de objetos sólo indican ausencia o presencia de un objeto, en el cuadrante correspondiente, por medio de los valores 0 ó 1. La detección de los objetos se realiza a cualquier distancia, siempre que no se interponga ningún obstáculo que impida su visualización. Los cuatro sensores afectados a la detección de un objeto, son seteados a 1, en el momento en que el objeto correspondiente es recogido.

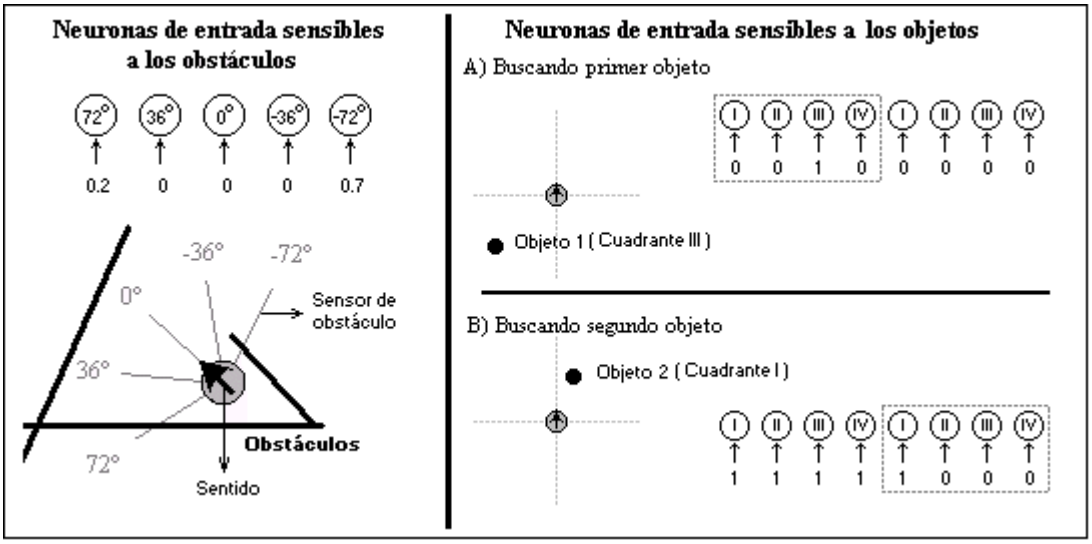


Figura 6. Agente

5.3. Escenarios definidos

La complejidad del problema varía en función de los distintos escenarios. El limitado alcance de visión para los obstáculos dificulta la movilidad del agente en ambientes amplios, con obstáculos alejados entre sí. Además, los escenarios han sido construidos para que una vez recogido el objeto

1, la estrategia de movimientos deba cambiar radicalmente para hallar el objeto 2, lo que dificulta aún más su control.

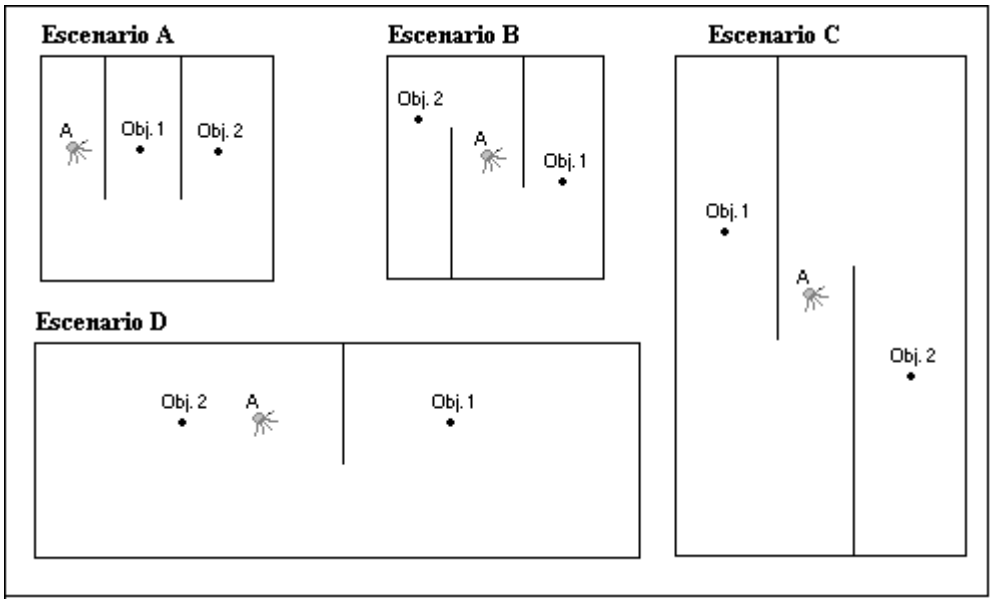


Figura 7. Escenarios

6. Detalles de implementación

6.1. Asignación del fitness

El problema de evasión de obstáculos puede considerarse del tipo *Sequential decision tasks*. Su característica principal es la dificultad para asignar con precisión la bondad de una acción tomada, siendo necesaria una secuencia de decisiones antes de poder medir cuál ha sido el efecto de cualquiera de ellas. Son ejemplos del mundo real: el enrutado de información en los routers de la Internet, el control de flujo químico en un reactor químico, el control de tráfico aéreo, etc. En todos estos casos, el efecto de una decisión simple se evidencia transcurrido un lapso de tiempo, y aún así, frecuentemente es difícil establecer cuáles decisiones fueron las responsables, y en qué medida, de lo acontecido[9].

Se ha enmarcado a la evasión de obstáculos y búsqueda de objetos dentro de este tipo de problemas y por lo tanto, no se hace ninguna valoración de aptitud de los controladores hasta que no haya concluido la simulación utilizada para su evaluación.

Una vez finalizado el tiempo de simulación se calcula el fitness del controlador que dirige al agente de la siguiente manera: Sea $f(a)$el valor de aptitud asignado al agente a por su desempeño en la simulación.

- Si el objeto 1 no fue alcanzado, $f(a)$ tomará un valor del intervalo $[0,40)$ calculado proporcionalmente al camino recorrido hacia dicho objeto.
- Si, por el contrario, el objeto 1 fue recolectado pero aún el objeto 2 no lo fue, $f(a)$ tomará un valor del intervalo $[40,50)$ calculado proporcionalmente al camino recorrido hacia el objeto 2.

- Finalmente en caso de haber recogido ambos objetos $f(a)$ pertenecerá al intervalo $[50,60)$. Si t es el número de instantes utilizados para completar la tarea, y T el número total de instantes de la simulación entonces, para este último caso, el fitness se calcula como $f(a)=50+10(T-t)/T$. De esta forma se obtiene un valor entre 50 y 60 proporcional a la velocidad con que se ha completado la tarea requerida.

6.2. Arquitectura de las redes neuronales artificiales

Para la conformación de los controladores se utilizaron redes feedforward con una única capa oculta formada por 8 neuronas, con esquema de conexión libre (no completamente conectadas), con término de tendencia y evolución de función de transferencia pudiendo cada nodo poseer una de entre cuatro sigmoides distintas: $f_1(x)=1/(1+\exp(-0.5x))$, $f_2(x)=1/(1+\exp(-x))$, $f_3(x)=1/(1+\exp(-1.5x))$, $f_4(x)=1/(1+\exp(-2x))$. La evolución de función de transferencia ha mostrado un buen rendimiento al ser aplicadas a los problemas de evasión de obstáculos [2][3], por ello ha sido utilizada en el presente trabajo.

7. Resultados obtenidos

El rendimiento de los controladores obtenidos a partir de esta nueva estrategia evolutiva, fue medido y comparado con SANE, sobre diversos escenarios de distintas complejidades. Se realizaron pruebas evolucionando arreglos de 3 y 4 componentes.

En todas las pruebas realizadas se evolucionaron redes feedforward con 13 neuronas de entrada, 2 de salida (3 en el caso de los arreglos neuronales), y 8 neuronas ocultas, conexión de tendencia y evolución de función de transferencia. En la población de neuronas se codificaron el tipo de sigmoide (2 bits), el peso de la conexión de tendencia (16 bits) y 15 conexiones por neurona. Cada conexión se codificó con 8 bits para la etiqueta y 16 bits para el peso. Los cromosomas de las subpoblaciones de blueprints se codificaron con números reales. Las evoluciones se prolongaron por 200 generaciones, culminando el proceso evolutivo aún en el caso de no lograr el objetivo propuesto en el escenario (recoger ambos objetos).

Se utilizaron poblaciones de 80 blueprints y 640 neuronas, tanto en SANE como en las subpoblaciones de ANES.

Para el algoritmo evolutivo del método ANES, se consideró que la curva del fitness se hace estacionaria al no mejorar durante 10 generaciones consecutivas (5% del número total de generaciones), permitiendo que cada subpoblación se evolucione varias veces a lo largo del proceso evolutivo. Una vez concluido un período evolutivo, se conservó el 25% de las mejores redes de la subpoblación para el próximo turno de evolución.

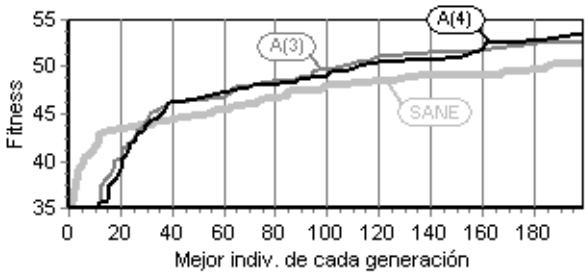
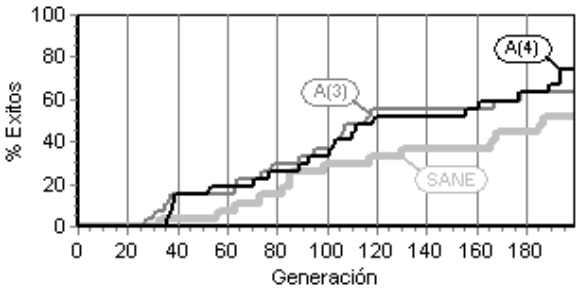
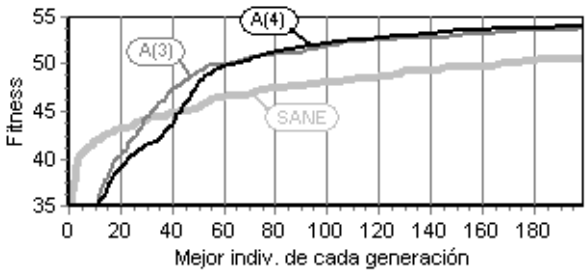
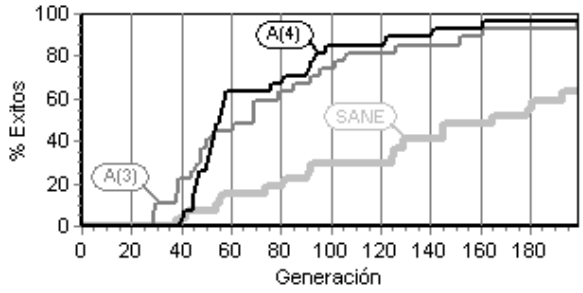
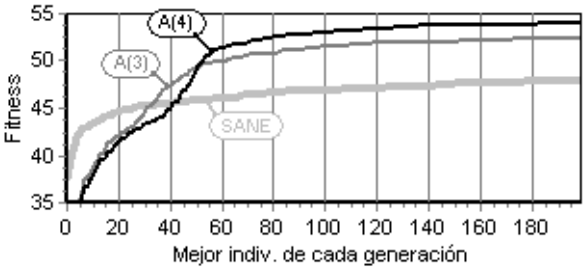
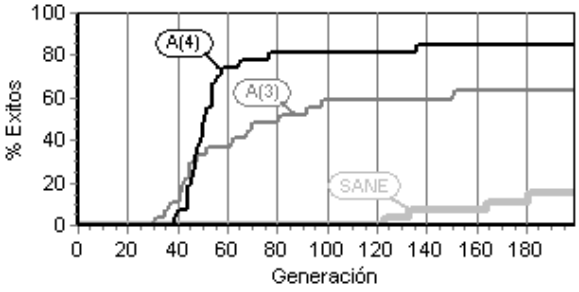
Se realizaron 50 evoluciones sobre los escenarios graficados en la figura 7 para cada método neuroevolutivo. De esta forma, se obtuvieron las curvas del mejor fitness por generación para cada método promediado entre las 50 evoluciones (figuras 8, 10, 12 y 14).

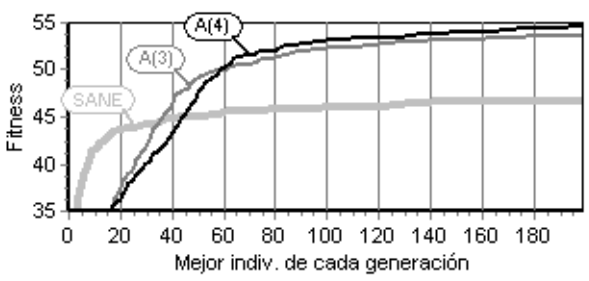
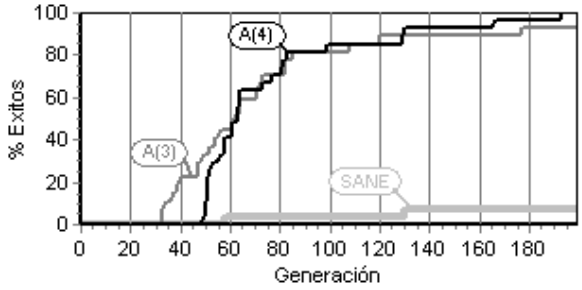
El porcentaje de evoluciones exitosas alcanzadas en las 50 realizadas, donde se consigue un controlador que comanda correctamente al agente en la resolución de la tarea impuesta, también es un dato importante que caracteriza el rendimiento de los métodos testeados. Con estos resultados, es posible estimar la probabilidad de éxito al correr una evolución utilizando SANE o ANES con

arreglos de 3 y 4 componentes (Figuras 9,11,13 y 15).

Las curvas del mejor fitness obtenido por generación evidencia el mejor desempeño de ANES en todos los escenarios testeados, obteniendo siempre los valores más altos.

Los datos recolectados sobre el porcentaje de evoluciones exitosas (en la que el objetivo del problema es efectivamente alcanzado) sobre las 50 realizadas, favorecen ampliamente a los arreglos de 3 y 4 componentes obtenidos con ANES. Esta relación es aún más ventajosa a la del fitness promedio. De aquí se deduce que existe mayor probabilidad de alcanzar el objetivo por medio de una evolución de arreglos neuronales que por medio de SANE. La diferencia de rendimiento de ambos métodos se hace particularmente notoria en los ensayos realizado sobre los escenarios C, y más aún sobre el D, donde se consiguen casi el 100% de éxitos utilizando ANES mientras que SANE alcanza el objetivo menos del 10% de las veces (figura 15).

	Promedio de fitness	**Porcentaje de éxitos obtenidos**
Escenario A	 **Figura 8**	 **Figura 9**
Escenario B	 **Figura 10**	 **Figura 11**
Escenario C	 **Figura 12**	 **Figura 13**

Escenario D	 **Figura 14**	 **Figura 15**

8. Conclusiones y líneas de trabajo futuras

Se ha presentado un método que evoluciona arreglos neuronales para controlar procesos en forma más eficiente, pero con similares requerimientos de almacenamiento y carga computacional que los métodos convencionales.

ANES también mejora otras estrategias de evolución de arreglos neuronales eliminando la definición explícita de subobjetivos, logrando así soluciones más fáciles de generalizar a otros tipos de problemas.

En una etapa futura, se propone estudiar el comportamiento de ANES con arreglos de longitud variable para que la evolución ajuste de manera automática el número de redes neuronales más adecuado para el controlador.

Se espera aplicar los resultados obtenidos hasta el momento en el área de la robótica para evolucionar arreglos neuronales que comanden un brazo robot de cinco grados de libertad a partir de un sistema de adquisición de imágenes externo.

Referencias

[1] Bruce, J. and Miikkulainnen, R. Evolving Populations Of Expert Neural Networks. Department of Computer Sciences, The University of Texas at Austin. *Proceedings of the Genetic and Evolutionary Computation Conference*. (GECCO-2001, San Francisco, CA), (2001), pp. 251--257.

[2] Corbalán, L., Pisano, M., Osella Massa, G. y Lanzarini L. Criaturas Virtuales especificadas a través de Redes Neuronales Evolutivas. *VII Congreso Argentino de Ciencias de la Computación*. Argentina, Vol. 2, (Octubre 2001), pp. 1105-1115.

[3] Corbalán Leonardo. Evolución de redes neuronales para comandar criaturas que alcanzan objetivos sorteando obstáculos en un entorno virtual 2D. *Tesis de grado correspondiente a la carrera Lic. en Informática*. UNLP. Marzo 2002.

[4] Corbalán Leonardo y Lanzarini Laura. Arreglos neuronales evolutivos aplicados a la evasión de obstáculos y alcance de objetivos. *VIII Congreso Argentino de Ciencias de la Computación CACIC 2002* (octubre 2002) - *XXVIII Conferencia Latinoamericana de Informática CLEI 2002* (noviembre 2002).

[5] Corbalán Leonardo y Lanzarini Laura. An ENA-Based Strategy Replacing Subobjectives Definition in Incremental Learning. *International Conference on Computer Science, Software Engineering, Information Technology, e-Business, and Applications, (CSITeA'03).* Junio 2003, Río de Janeiro, Brasil.

[6] Freeman, J. A. & Skapura, D. M. Redes neuronales Algoritmos, aplicaciones y técnicas de programación. Addison–Wesley, 1991. Versión en español de: Rafael García -Bermejo Giner. Addison–Wesley Iberoamericana 1993.

[7] Goldberg, D. E. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison–Wesley, 1989. pág. 10

[8] Moriarty, D. E. & Miikkulainen, R. Efficient Reinforcement Learning through Symbiotic Evolution. Department of Computer Sciences, The University of Texas at Austin. Austin, TX 78712. *Machine Learning*, Vol. 22, (1996), pp.11-33.

[9] Moriarty, D. E. Symbiotic Evolution of Neural Networks in Sequential Decision Tasks. *Department of Computer Sciences, The University of Texas at Austin Ph.D. Dissertation. Technical Report AI97*-257, January 1997.

[10] Moriarty, D. E. & Miikkulainen, R. Forming Neural Networks Through Efficient and Adaptive Coevolution. Information Sciences Institute, University of Southern California. Department of Computer Sciences, The University of Texas at Austin. *IEEE Transactions on Evolutionary Computation*. Vol.5, (1997), pp.373-399.

[11] Yao, X. and Liu, Y. Ensemble Structure of Evolutionary Artificial Neural networks. *Computational intelligence Group, School of Computer Sciencie University College*. Australian Defence Force Academy, Canberra, ACT, Australia 2600. 1996.

[12] Yao, X. and Liu, Y. A New Evolutionary System for Evolving Artificial Neural Networks. *IEEE Transactions on Neural Networks*, Vol 8, nro. 3, pp 694-713, 1997

[13] Yao, X. Evolving Artificial Neural networks. School of Computer Sciencie The University of Birmingham Edgbaston, Birmingham B15 2TT. *Proceedings of the IEEE*. Vol.87, No.9, (September 1999), pp.1423-1447.